

Provisional Patent Application
Docket No. 9339/31029

5

10

15

20

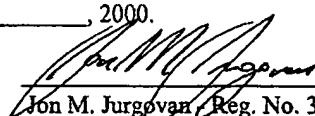
Be it known that I, John David West Brothers, a citizen of the United States of America, residing at 1620 Grand Junction, Alpharetta, GA, 30004-7842, have conceived an invention entitled:

25 **SYSTEM AND METHOD FOR THE MANAGEMENT OF SECURE CONTENT ON THE
HTTP DISTRIBUTION INFRASTRUCTURE**

of which the following is a specification:

30

I hereby certify under 37 C.F.R. § 1.10 that this correspondence is being deposited with the United States Postal Service with sufficient postage for Express Mailing, mailing label no. EL627102543US in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231, on August 11, 2000.


Jon M. Jurgovan Reg. No. 34,633

System And Method for the Management of Secure Content on an HTTP Distribution Infrastructure

Abstract

A URL encoding and decoding system whereby web servers are able to host, retrieve and control access to web-hosted data files across a distributed web-content infrastructure. The primary website provider is able to generate specialized URL links in real time based on the access rights of the visiting user, and encode those access rights and user proof-of-identity into the URLs that are presented on standard web pages. These specialized encodings allow the primary website provider to limit access to the content based on a variety of factors, including the user's IP address, the time of day, the frequency of access and other factors. When the user accesses the URL to retrieve the data, the web server receiving the request is able to examine the request, and validate or deny the user's request based on the comparison of the data encoded into the URL with the manifest data about the user and the environment. If the user's request is validated, the web server is able to either deliver the requested data immediately, or, if the data is unavailable, use the information derived from the encoded URL to create its own specialized request to retrieve the data from another web server which may possesses a copy of the requested data.

Background of the Invention

A now-common service on the Internet is content distribution, which allows web sites to "publish" common data files to a wide range of caching servers, which allow users to access the content from a nearby, lightly loaded server, instead of from the original, centralized, heavily loaded server. The primary advantage of the content distribution system is that it allows a web content provider to absorb significantly more traffic, since the traffic is shared across multiple web servers. One of the primary weaknesses of this model is the inability of the content provider to distribute files intended for only limited access. Because of the nature of these content distribution systems, it is difficult, if not impossible to ensure that only a select group of users are able to access the content. This, in turn, makes it difficult for subscription-based websites, or websites with subscription or limited access content to be able to publish that content using a content distribution network.

This invention is novel and valuable in that it allows this limited access data to be published into a distribution network, while still ensuring that only a limited audience is able to view the content.

Description

This system consists of several subcomponents, which work together in three phases. The first phase, provisioning, allows a content provider to send a list of encryption keys and encryption key controls to the appropriate web servers that will operate as elements of the secure content management system. These keys and controls are stored in the **Secure Content Key Database** on the appropriate web servers. Once these keys have been defined, the system is able to move into the next phase, which is operations

In the operations phase, the initial component that starts the secure transfer of data is the **Secure URL Generator**, which operates at a 'primary' web site, (a website that is responsible for the management of content and access rights to that content). This **Secure URL Generator** creates **Secure URLs**, which are URLs that are specially encoded to prove that the user attempting to retrieve the content from any remote web server is able to do so, as long as all access rights are valid, and the user's identity can be ensured. This proof of identity and the proof of access rights are protected from modification through the use of encryption keys held within a **Secure Content Key Database**. When the user activates the **Secure URL**, the data request travels to a **Rights Management Enforcer**, which is a subcomponent of a web server maintained within a content distribution network. The **Rights Management Enforcer** is able to examine the **Secure URL**, to determine if the requesting user ip address is the same as the user ip address allowed to access the content as specified by the **Secure URL Generator**, to verify that the encryption keys and digital signatures specified within the **Secure URL** match or are derivable to the elements specified in the **Secure Content Key Database** and that the various access rights associated with the **Secure URL** are still valid. If the access rights are valid, and the user's ip address is valid, the **Rights Management Enforcer** passes the request onto the **Secure Caching System**. The **Secure Caching System** examines the request, and determines if the data file is available on the local web server, using arbitrary methods. If the data file is available on the local web server, the **Secure Caching System** can choose to deliver the file to the requesting user, updating any access rights elements necessary. If the data file is not available on the local web server, the **Secure Caching System** determines what remote server to retrieve the data file from, or how to retrieve the file if it should be retrieved in some other way, and, if necessary, uses an appropriate **Secure URL Generator** and **Secure Content Key Database** to generate its own **Secure URL** to request the data from the specified remote server. The **Secure Caching System** then may use that **Secure URL** to request data from the remote server. The remote server in turn, may or may not use its own **Rights Management Enforcer** and **Secure Content Key Database** to validate the request as described above. The remote server's **Secure Caching System** may deliver the file, or may itself generate a secure or insecure URL request or other data to another web server. This process may continue indefinitely.

The third (optional) phase is maintenance, where encryption keys and controls are added to and removed from the secure content key database as desired by the content provider, using arbitrary methods that are beyond the scope of this invention.

Applications of this Invention

Applications of this invention include the primary application – ensuring that users who request data files have been granted to receive the file. Additional applications include, but are not limited to:

- 1) Using access rights management to ensure that a particular data file is requested a limited number of times before the access rights expire.
- 2) Using access rights management to ensure that requests for a particular data file fall within a specified time frame.
- 3) Using user identity encoding to track which user accessed the data, to build up demographic maps
- 4) Using the IP address matching system to ensure that a limited set of IP addresses are allowed to access the content

Subcomponent Descriptions

Secure Content Key Database

The secure content key database is a data file in arbitrary form, which is hosted on a web server or web server system. This database contains a list of one or more rows of data. Each row consists of one encryption key and may or may not contain description of the controls and limits associated with each key. Each row must also be uniquely associated with a specific content provider, which may be done by the use of data within the row that corresponds to the name of the content provider's web site, or by maintaining multiple databases, each associated with one content provider, a combination of these two, or other arbitrary methods that allow each row to be uniquely associated with one specific content provider.

With the addition of new elements to each row, the secure content key database can grow in functionality. Examples include:

- Dividing the keys in the database into two groups, one of which is used to validate client requests, the other of which is used by the web server to retrieve content from another web server.
- Adding a start and/or end date associated with each key, which prevents keys from being used before a certain date or after a certain date.
- Adding an IP address or name of a remote content server, which should be used to retrieve the content specified in the secure URL.
- Adding an index value, which can be used by the secure URL to specify a given key.
- Adding a 'retrieval key' to each row, which is used by the web server to retrieve content from another web server.
- Adding a 'hashing algorithm' identifier to each row, which is used by the Rights Management Enforcer to determine how to perform the cryptographic validation of the secure URL.

Secure URL Generator

The secure URL generator is responsible for creating secure URLs from basic URLs and various additional rights-management elements. A secure URL can be created in many different forms, as long as the appropriate data elements are properly represented within the URL itself. Two format examples include the “formatted path” approach, whereby the URL is modified to include secure data elements as part of the file path to the desired content, and the “appended argument” approach, whereby the URL is modified to include secure data elements appended to the end of the URL.

Example of “formatted path”

Original URL:

http://www.content-server.com/path1/path2/file.ext

becomes

http://www.content-server.com/secure_content_data/path1/path2/file.ext

Example of “appended argument”:

Original URL:

http://www.content-server.com/path1/path2/file.ext

becomes

http://www.content-server.com/path1/path2/file.ext?secure_content_data

Note that each of these is simply an example of what could be done. The exact nature of the encoding of the URL is not relevant to this invention, as long as the secure content data is integrated into the URL.

URL Components

A URL consists of the following components:

Header	Destination	Data Request Fields
http://	www.content-server.com	/path1/path2/file.ext

Data

The secure content data consists of at least the following elements:

- The IP address of the user, in some form (for example, as an integer value represented in string form, in standard ‘dotted quad’ notation in string form, or other formats that retain all or significant portions of the IP address information)

- An encrypted hash value, that represents a mathematical combination of the IP address, some or all of the path and file elements of the original URL, and the encryption key.
 - This hash value may be calculated using any number of cryptographic methods, including a one-way hashing algorithm, such as SHA-1 or MD5, or a two-way encrypting algorithm, such as DES or 3-DES. The exact nature of the algorithm used is not relevant to the patent, as long as the following conditions are held:
 - The hashing algorithm consistently calculates the same hash value based on the same input data
 - It is possible for multiple computers using the hashing algorithm to calculate the same hash value based on the same input data.
 - The action of “forging” the hash value, to allow the URL to be modified by the user before the request requires cryptographic analysis – in other words, it isn’t a trivial problem to solve.

Some examples of how this hash value could be created include:

One Way Hash

In this approach, all of the important components of a secure URL are generated, except for the final hash value. The encryption key is appended to the intermediate URL, which is hashed using a one way hashing algorithm such as SHA-1. The resulting hash value is appended to the intermediate URL, replacing the encryption key, and the result is the secure URL.

There are many ways to handle this encoding of the URL, the following example is one way. Persons of normal skill in the art will recognize that the exact encoding format is irrelevant as long as the following things are true:

- Both the Secure URL Generator and the Rights Management Enforcer understand the same encoding format and encoding/decoding protocol.
- The IP Address and the Encryption Key (or some derivative of the encryption key) must make up components of the hashed data that is delivered to the one-way hashing algorithm
- The Encryption Key must not be available “in the clear” as part of the unencrypted data within the URL.

The key elements include:

- User IP Address: 1.2.3.4
- Insecure URL: http://www.content-provider.com/path/file.ext
- Encryption key: 100

Using the “appended argument” approach, a textual string could be crafted as follows:

String a: http://www.content-provider.com/path/file.ext?ip=1.2.3.4&hash=

The encryption key is then appended to this string:

String b: http://www.content-provider.com/path/file.ext?ip=1.2.3.4&hash=100

This URL is then hashed, using a one-way hashing function, which generates an 8 byte hash value.

Hash Value C: acd54dcd3d8eca892acb34e4b5d1c8c

Hash Value C is then appended to String a:

String D: http://www.content-provider.com/path/file.ext?ip=1.2.3.4&hash=acd54dcd3d8eca892acb34e4b5d1c8c

Two Way Encryption

In this approach, all of the important components of a secure URL are generated, except for the final hash value. The encryption key, along with the intermediate URL are used as the arguments to a two-way encryption function, such as DES, which produces an encrypted version of the intermediate URL. The resulting encrypted data is appended to the intermediate URL, replacing the encryption key, and the result is the secure URL.

The key elements include:

- IP Address: 1.2.3.4
- Insecure URL: http://www.content-provider.com/path/file.ext
- Encryption key: 100

Using the “appended argument” approach, a textual string is crafted as follows:

String A: http://www.content-provider.com/path/file.ext?ip=1.2.3.4

This string and the encryption key are processed by a two-way encryption algorithm, such as the well-known DES encryption algorithm. The result is an encrypted value.

Encrypted Data B: da892acb3454dcd3d5d1c8ac8ece4bc

The Encrypted Data B is then appended to the Insecure URL, to create the Secure URL:

String C: http://www.content-provider.com/path/file.ext?data=da892acb3454dcd3d5d1c8ac8ece4bc

Additional Secure Content Data Fields

In addition to the required fields, numerous optional fields may also be available:

- Tracking Identifier
 - This represents a string of digits, letters, etc, that the content provider wishes to append to the secure content to help track which users are using the data and when.
- End Timestamp

- This represents an absolute last time in some arbitrary format for which this particular URL is considered “valid”. When a web server receives a request that occurs after the time represented by this end timestamp has occurred, the request will be considered invalid.
- Start Timestamp
 - This represents an absolute first time in some arbitrary format for which this particular URL is considered “valid”. When a web server receives a request that occurs before the time referenced by this start timestamp has occurred, the request will be considered invalid
- Lifespan
 - This represents a “lifespan” for the secure URL, indicating how long the URL is valid after the first request is generated. When a web server receives the first request with a given URL, it records the time of this initial request. When subsequent requests are generated using the same secure URL, if the time associated with the lifespan of the URL has expired, the request will be considered invalid.
- Maximum Reference
 - This represents the maximum number of times that the secure URL may be accessed. When a web server receives the first request with a given secure URL, the web server records the incident, and associates a counting variable with the URL. Subsequent requests with the secure URL increment the counter. Once the maximum number of requests has been reached, additional requests will be considered invalid.
- Key Index
 - This represents an identifier which can be used by the web server to determine which encryption key should be used to validate the secure URL. If present, the web server will examine the elements in the Secure Content Key Database, to find the encryption key associated with the index. If the given index is invalid, or if the key is no longer valid, the request will be considered invalid.
- Format Field
 - This represents a description of which special additional identifiers are present in the secure URL, which reduces the complexity within the Rights Management Enforcer of evaluating the secure URL. It may also describe how various fields are encoded, for example, if the IP address is encoded using dotted quad notation, or integer notation.
- Encryption Model
 - This represents a description of what encryption strategy (one way, two way, etc) was used to create the encrypted hash value.
- Encryption Algorithm
 - This represents a description of which algorithm was used to create the encrypted hash value

Note that this list is a list of example elements, and does not contain all possible elements which could be added to a secure URL.

Rights Management Enforcer

This process or service runs on the 'distributed' web server or as part of the distributed web server system. It is responsible for validating a http request that uses a secure URL, determining if the secure URL is properly formatted, contains the required data fields, can be arithmetically validated using the known encryption keys and is still considered a valid URL based on existing access rights that may or may not be encoded into the secure URL.

The RME takes the secure URL, and performs the following actions on it:

- 1) It cryptographically validates the encrypted hash value, either using a default algorithm, an algorithm specified in the secure URL, or an algorithm specified in the Secure Content Key Database.
- 2) Once the encrypted hash value has been validated, the various data fields are then accessible to the RME. The only field that must be validated is the IP address, which must be successfully matched to the IP address associated with the TCP stream associated with the request. This matching may either be done by an arithmetic comparison of integer values, or a string-based comparison of the string-encoded IP address, in an arbitrary format.

The other secure content data fields (if present) must also be examined at this point. If the secure content rights include a maximum number of requests, or a lifespan, the data associated with these rights must be stored on the target web server for later use. It is beyond the scope of this invention to discuss how these fields are managed and maintained.

If the secure URL is found to be invalid, the RME may choose how to proceed. It may choose to pass the request on to the Secure Caching System for delivery. It may choose to send various status responses to the requesting user, for example 404 (file not found) or 403 (forbidden). This is beyond the scope of this invention.

Secure Caching System

Once the RME has determined that the secure URL is valid, it passes the sub-elements of that secure URL associated with the requested data object to the Secure Caching System. The SCS is responsible for either responding to the request via standard models (such as delivering the requested file back to the requesting user, sending an unknown file response, sending some sort of 'default' file, sending a 'forbidden access' response or other response.) These responses are beyond the scope of this invention.

If the Secure Caching System determines that it wishes to deliver the file to the user, but the file is not located on the local file system, it may choose to retrieve the file from another server, for example, the primary web server for the content, or any other server that may or may not have the data.

If the Secure Caching System wishes to retrieve the data from a remote server, it may need to create its own secure URL to generate the appropriate response. In this case, the Secure Caching System uses the algorithms within the Secure URL Generator to create a

Secure URL of its own, that proves that the physical sever associated with the Secure Caching System is able to retrieve the data from a secure web server.

This Secure URL will almost always be different from the original Secure URL, since the IP address of the requesting server is most likely not going to be the same as the IP address of the originally requesting client.

It is also possible that various elements within the Secure Content Key Database dictate a different encryption key to use to generate the request. They may also specify a specific web server to generate the request to retrieve the desired data. These additional elements are beyond the scope of this invention.

Note that the same validation and generation steps used to generate and validate the original Secure URL can also be used to generate and validate this secondary Secure URL. Thus this process can be repeated numerous times within a single transaction. It is even possible, although inadvisable for these requests to be generated in a loop, which would ensure that the original request could never be filled.

If the request for data from the remote web server is rejected or ignored, the Secure Caching System may choose to send an alternate file, send various failure status codes, ignore the original request or other action. These actions are beyond the scope of this invention.

If the request for data from the remote web server is accepted, it is optionally possible for the data content to be encrypted during the transmission to the Secure Caching Server. This is beyond the scope of this invention.

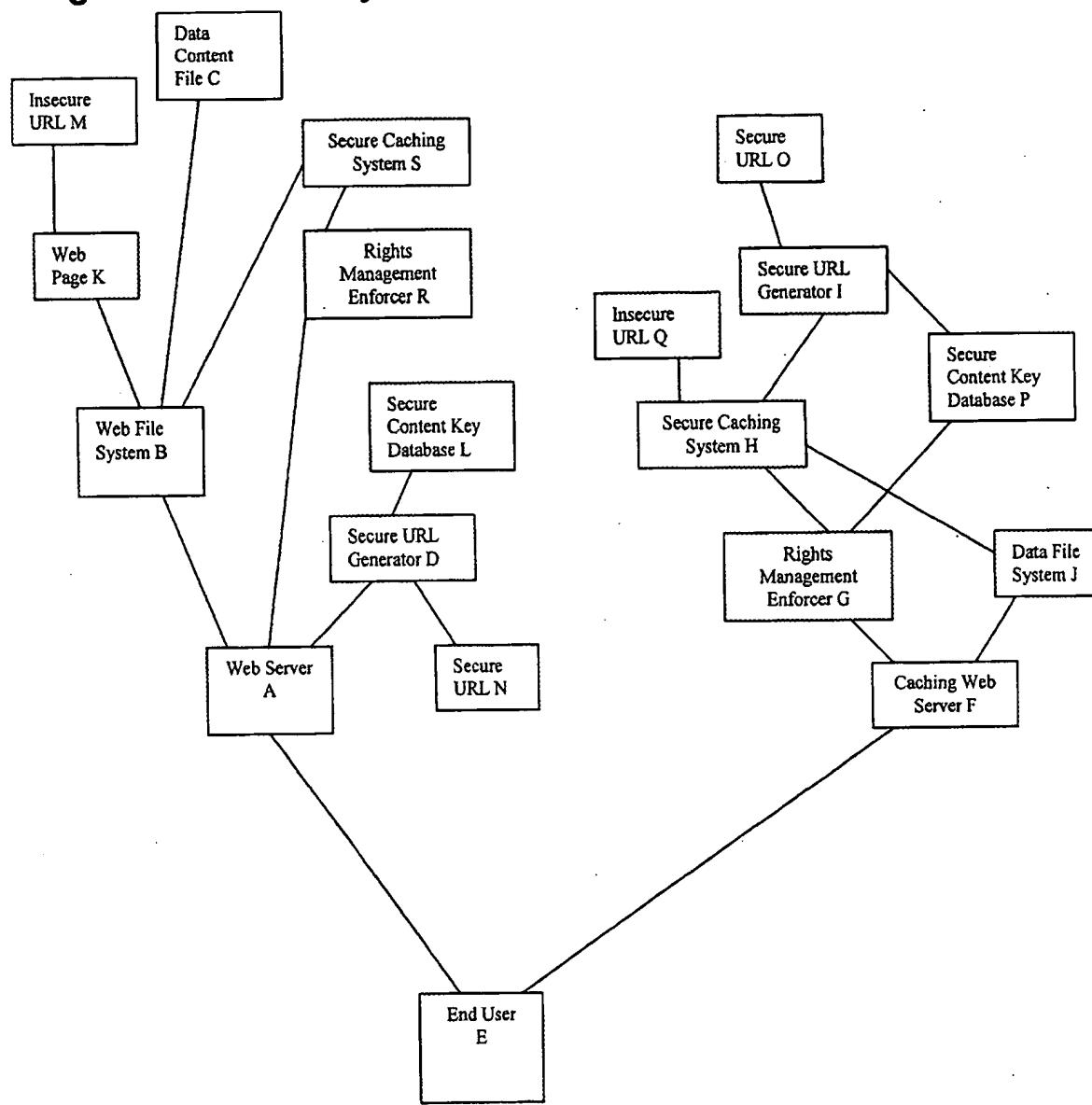
Once the requested data is received in unencrypted form on the local system by the Secure Caching System, the data may be delivered to the user using standard methods, although the Secure Caching System may choose to respond differently. The data may also optionally then be stored on the local caching system, using standard methods.

Once the transaction has been completed, the Secure Caching System may optionally generate logging data to historically record the data access.

Other Notes

This system does not necessarily require the use of the HTTP protocol. A similar system could be generated using the FTP protocol, or any number of other data request and retrieval protocols that operate across a distributed infrastructure, and are accessible using a URL-based protocol.

Diagram – Overall System



In this diagram, End User E sends an HTTP-protocol web page request to Web Server A, requesting Web Page K (an HTML document). Web Server A retrieves the requested Web Page K from Web File System B. Web Server A then requests Secure URL Generator D to create Secure URL N. Secure URL Generator D uses the encryption key data contained within Secure Content Key Database L, and the original Insecure URL M to generate the Secure URL N.

Web Server A replaces Insecure URL M with Secure URL N within Web Page K, using standard methods, such as a buffer copy or a web application server.

Web Server A then sends Web Page K, containing Secure URL N to End User E.

End User E receives Web Page K, and processes the data contained within. End User E then "activates" Secure URL N, using the standard method of examining the destination address field, generating a TCP-based communications connection to the machine referenced by this destination address field (in this case, Caching Web Server F), and sending the data request fields within Secure URL N to Caching Web Server F, using the HTTP protocol. In this case, the request contained within Secure URL N is for the data contained within Data Content File C.

Caching Web Server F receives the data request fields from within Secure URL N, and determines, using standard methods (for example, examining the string content with the data request fields for certain characters or words) that the request should be validated by Rights Management Enforcer G.

Rights Management Enforcer G receives the data request fields from within Secure URL N, and validates these fields, using encryption key data contained within Secure Content Key Database P. If Rights Management Enforcer G determines that the data request fields are valid, it passes these data request fields on to Secure Caching System H.

Secure Caching System H examines Web File System J to determine if Data Content File C is located there. If not, Secure Caching System H determines what other web server to retrieve Data Content File C from, using standard methods (such as using special information encoded into the URL request, or examining fields contained within Secure Content Key Database P). In this case, it is determined that Web Server A is the target web server.

Secure Caching System H then creates Insecure URL Q, which allows it to communicate with Web Server A, and sends Insecure URL Q to the Secure URL Generator I. Secure URL Generator I uses the data within Secure Content Key Database P and the information in Insecure URL Q to generate Secure URL O.

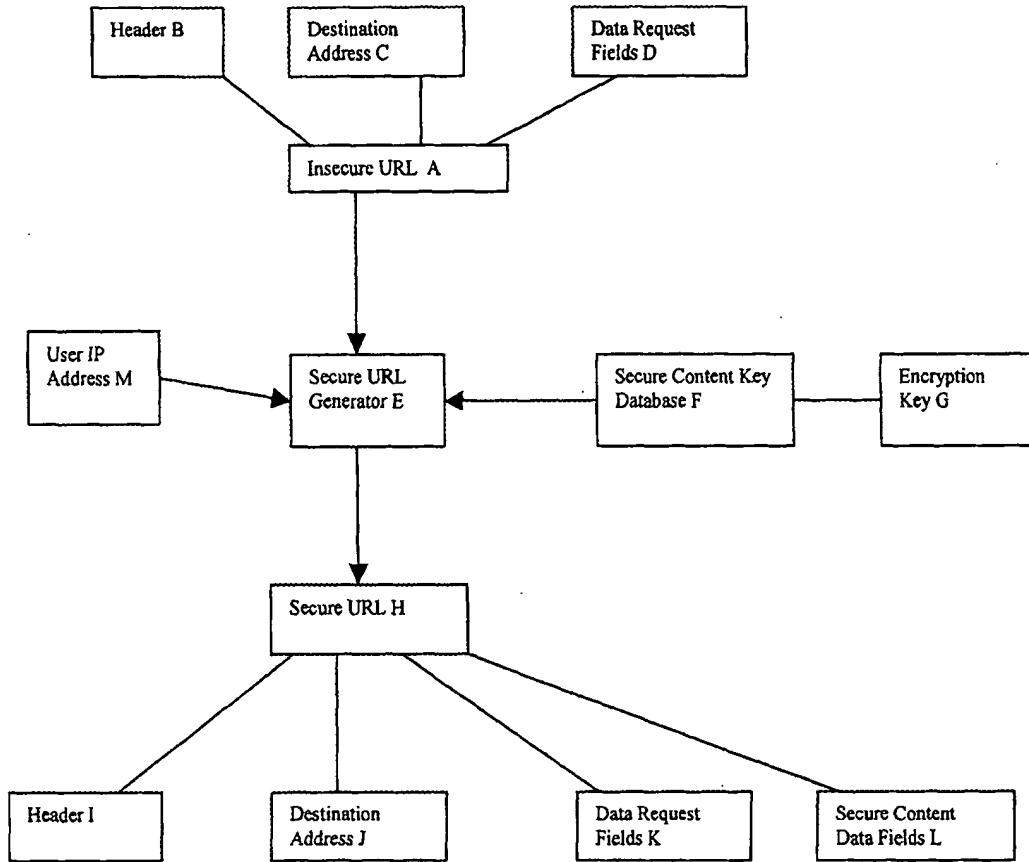
Secure Caching System H then uses Secure URL O to create an HTTP request to Web Server A, using standard methods. Web Server A receives the data request fields from within Secure URL O and determines, using standard methods, that the request should be validated by Rights Management Enforcer R.

Rights Management Enforcer R receives the data request fields from within Secure URL O, and validates these fields, using encryption key data contained within Secure Content Key Database L. If Rights Management Enforcer R determines that the data request fields are valid, it passes these data request fields on to Secure Caching System S.

Secure Caching System S examines Web File System B to determine if Data Content File C is located there. If so, Secure Caching System S generates an HTTP response message for Secure URL O, which contains the data from within Data Content File C. This data is received by Secure Caching System H, which writes a copy of the data to Web File

System J and also generates an HTTP response message for Secure URL N, which contains the data from within Data Content File C. This response message is delivered back to End User E.

Diagram – The Creation of a Secure URL



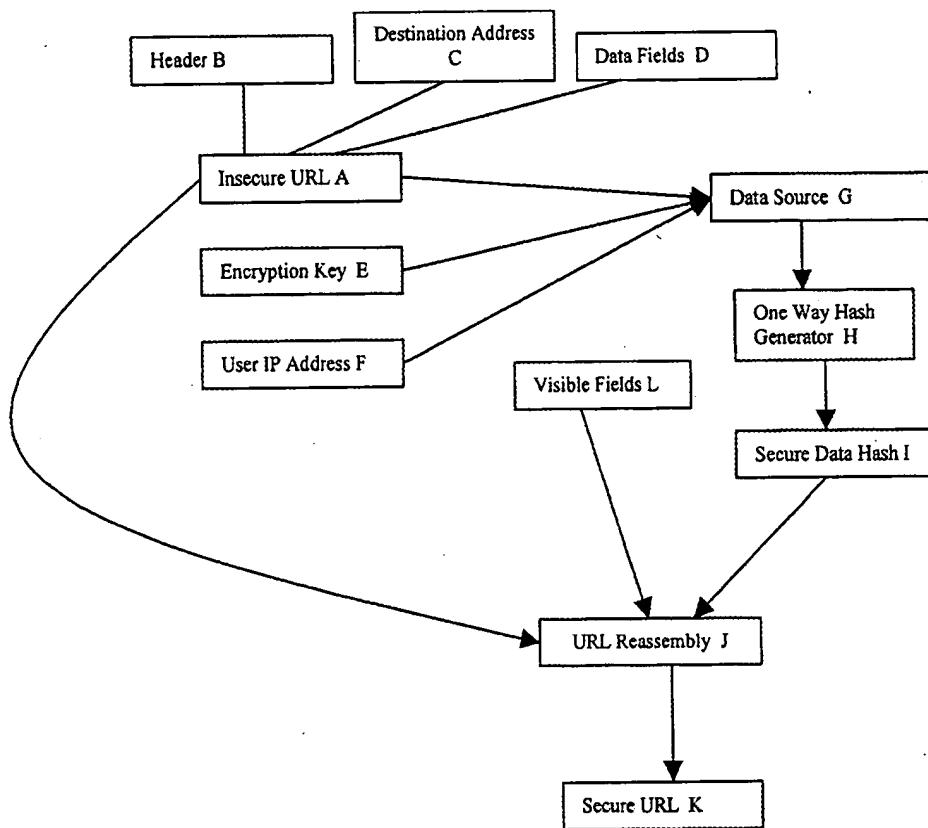
In this diagram, Secure URL Generator E takes Insecure URL A, and converts it into Secure URL H. It executes the following process:

Secure URL Generator E receives Insecure URL A as a data element. Insecure URL A consists of the following 3 data fields: Header B (for example, the string “<http://>”) Destination Address C (for example, the string “www.incanta.net”) and Data Request Fields D (for example, the string “/path1/path2/file.doc”). Note that it is not necessary for Insecure URL A to be a “viable” URL.

Secure URL Generator E accesses Secure Content Key Database F, and determines which encryption key should be used, using arbitrary methods. Choosing Encryption Key G, Secure URL Generator E then combines the data associated with Insecure URL A, User IP Address M, and Encryption Key G to create Secure URL H. Secure URL H consists of Header I, which may or may not be the same as Header B, Destination

Address J which may or may not be the same as Destination Address B, Data Request Fields K which may or may not be the same as Data Request Fields C, and the new element: Secure Content Data Fields L, which consists of the mathematically manipulation of the data contained in Data Request K, User IP Address M, and Encryption Key G, at a minimum, and may consist of other data fields as well (including ones not shown in this diagram).

Example Diagram of the Creation of the Secure Content Data Fields



In this diagram, as an example, the minimal set of data components are combined using a specific hashing algorithm to create a Secure URL of a specific type.

Using simple string manipulation tools (such as the `strcat` and the `sprintf` functions in C/C++), Data Fields D, Encryption Key E and User IP Address F are combined together into a single data element Data Source G.

Data Source G is then provided as the input to One Way Hash Generator, which creates Secure Data Hash I using any arbitrary one-way hash algorithm (such as SHA-1 or MD5) which produces a hash value, and encoding the resulting hash value in a string format using various well-known methods, such as hexadecimal encoding to convert data bytes into pairs of hexadecimal string-encoded digits.

Visible Fields L is also created, and consists of the following text: "?poi=" (Note that the exact text used is not relevant to the invention, nor is this field necessary in some implementations of the secure URL).

URL Reassembly J takes the data from Insecure URL A, Visible Fields L and Secure Data Hash I and combines them together using well known string manipulation functions such as `strcat` and `sprintf`, creating Secure URL K. In this example, it concatenates the fields together as follows: Insecure URL A, Visible Fields L and Secure Data Hash I. Example data:

If Insecure URL A is: "<http://www.incanta.net/path1/path2/file.ctx>", Visible Fields L is "?poi=" and Secure Data Hash I is 45abc3458f948d9b2c3d4d57a6cd675e87823bc8
Then Secure URL K would be:
<http://www.incanta.net/path1/path2/file.ext?poi=45abc3458f948d9b2c3d4d57a6cd675e87823bc8>